A polynomial time algorithm for computing the HNF of a module over the integers of a number field

Jean-François Biasse
Departement of computer science
2500 University Drive NW
Calgary Alberta T2N 1N4
biasse@lix.polytechnique.fr

Claus Fieker
Fachbereich Mathematik
Universität Kaiserslautern
Postfach 3049
67653 Kaiserslautern - Germany
fieker@mathematik.uni-kl.de

ABSTRACT

We present a variation of the modular algorithm for computing the Hermite Normal Form of an $\mathcal{O}_K$-module presented by Cohen [2], where $\mathcal{O}_K$ is the ring of integers of a number field K. The modular strategy was conjectured to run in polynomial time by Cohen, but so far, no such proof was available in the literature. In this paper, we provide a new method to prevent the coefficient explosion and we rigorously assess its complexity with respect to the size of the input and the invariants of the field K.

Categories and Subject Descriptors

I.1.2 [**Algorithms**]: Algebraic algorithms—*Symbolic and Algebraic Manipulation*

General Terms

Theory, Algorithms

Keywords

Hermite Normal Form, Complexity, Modules, Number theory

1. INTRODUCTION

The construction of a good basis of an $\mathcal{O}_K$-module, where K is a number field and $\mathcal{O}_K$ its ring of integers, has recently received a growing interest from the cryptographic community. Indeed, $\mathcal{O}_K$-modules occur in lattice-based cryptography [8, 9, 10, 13, 14], where cryptosystems rely on the difficulty to find a short element of a module, or solving the closest vector problem. The computation of a good basis is crucial for solving these problems, and most of the algorithms for computing a reduced basis of a $\mathbb{Z}$-lattice have an equivalent for $\mathcal{O}_K$-modules. However, applying the available tools over $\mathbb{Z}$ to $\mathcal{O}_K$-modules would result in the loss of of their structure.

The computation of a Hermite Normal Form (HNF)-basis was generalized to $\mathcal{O}_K$-modules by Cohen [2, Chap. 1]. His algorithm returns a basis that enjoys similar properties as the HNF of a $\mathbb{Z}$-module. A modular version of this algorithm is conjectured to run in polynomial time, although this statement is not proven (see last remark of [2, 1.6.1]). In addition, Fieker and Stehlé's recent algorithm for computing a sized-reduced basis relies on the conjectured possibility to compute an HNF-basis for an $\mathcal{O}_K$-module in polynomial time [5, Th. 1]. This allows a polynomial time equivalent of the LLL algorithm preserving the structure of $\mathcal{O}_K$-module. In this paper, we adress the problem of the polynomiality of the computation of an HNF basis for an $\mathcal{O}_K$-module by presenting a modified version of Cohen's algorithm [2, Chap. 1]. We thus assure the validity of the LLL algorithm for $\mathcal{O}_K$-modules of Fieker and Stehlé [5] which has applications in lattice-based cryptography, as well as in representations of matrix groups [4] and in automorphism algebras of Abelian varieties. In addition, our HNF algorithm allows to compute a basis for the intersection of $\mathcal{O}_K$ modules, which has applications in list decoding codes based on number fields (see [6] for their description).

Our contribution. We present in this paper the first polynomial time algorithm for computing an HNF basis of an $\mathcal{O}_K$-module based on the modular approach of Cohen [2, Chap. 1]. We rigorously adress its correctness and derive bounds on its run time with respect to the size of the input, the dimension of the module and the invariants of the field.

2. GENERALITIES ON NUMBER FIELDS

Let K be a number field of degree d. It has $r_1 \leq d$ real embeddings $(\sigma_i)_{i\leq r_1}$ and $2r_2$ complex embeddings $(\sigma_i)_{r_1<i\leq 2r_2}$ (coming as r_2 pairs of conjugates). The field K is isomorphic to $\mathcal{O}_K \otimes \mathbb{Q}$ where $\mathcal{O}_K$ denotes the ring of integers of K. We can embed K in

$$K_{\mathbb{R}} := K \otimes \mathbb{R} \simeq \mathbb{R}^{r_1} \times \mathbb{C}^{r_2},$$

and extend the σ_i's to $K_{\mathbb{R}}$. Let T_2 be the Hermitian form on $K_{\mathbb{R}}$ defined by

$$T_2(x, x') := \sum_i \sigma_i(x)\overline{\sigma_i}(x'),$$

and let $\|x\| := \sqrt{T_2(x, x)}$ be the corresponding L_2-norm. Let $(\alpha_i)_{i\leq d}$ such that $\mathcal{O}_K = \oplus_i \mathbb{Z}\alpha_i$, then the discriminant of K is given by $\Delta_K = \det^2(T_2(\alpha_i, \alpha_j))$. The norm of an element $x \in K$ is defined by $\mathcal{N}(x) = \prod_i |\sigma_i(x)|$.

To represent $\mathcal{O}_K$-modules, we rely on a generalization of the

notion of ideal, namely the fractional ideals of $\mathcal{O}_K$. They can be defined as finitely generated $\mathbb{Z}$-modules of K. When a fractional ideal is contained in $\mathcal{O}_K$, we refer to it as an integral ideal, which is in fact an ideal of $\mathcal{O}_K$. Otherwise, for every fractional ideal I of $\mathcal{O}_K$, there exists $r \in \mathbb{Z}_{>0}$ such that rI is integral. The sum and product of two fractional ideals of $\mathcal{O}_K$ is given by

$$IJ = \{i_1 j_1 + \cdots + i_l j_l \mid l \in \mathbb{N}, i_1, \cdots i_l \in I, j_1, \cdots j_l \in J\}$$
$$I + J = \{i + j \mid i \in I, j \in J\}.$$

The fractional ideals of $\mathcal{O}_K$ are invertible, that is for every fractional ideal I, there exists $I^{-1} := \{x \in K \mid xI \subseteq \mathcal{O}_K\}$ such that $II^{-1} = \mathcal{O}_K$. The set of fractional ideals is equipped with a norm function defined by $\mathcal{N}(I) = \det(M^I)/\det(\mathcal{O}_K)$ where the rows of M^I are a $\mathbb{Z}$-basis of I. The norm of ideals is multiplicative, and in the case of an integral ideal, we have $\mathcal{N}(I) = |\mathcal{O}_K/I|$. Also note that the norm of $x \in K$ is precisely the norm of the principal ideal $(x) = x\mathcal{O}_K$. Algorithms for ideal arithmetic in polynomial time are described in Section 5.

3. THE HNF

Let $M \subseteq K^l$ be a finitely generated $\mathcal{O}_K$-module. As in [2, Chap. 1], we say that $[(a_i), (\mathfrak{a}_i)]_{i \leq n}$, where $a_i \in K$ and $\mathfrak{a}_i$ is a fractional ideal, is a pseudo-basis for M if

$$M = \mathfrak{a}_1 a_1 \oplus \cdots \oplus \mathfrak{a}_n a_n.$$

Note that a pseudo-basis is not unique, and the main result of [5] is precisely to compute a pseudo-basis of short elements. If the sum is not direct, we call $[(a_i), (\mathfrak{a}_i)]_{i \leq n}$ a pseudo-generating set for M. Once a pseudo-generating set $[(a_i), (\mathfrak{a}_i)]_{i \leq n}$ for M is known, we can associate a pseudo-matrix $A = (A, I)$ to M, where $A \in K^{n \times l}$ and $I = (\mathfrak{a}_i)_{i \leq n}$ is a list of n fractional ideals such that

$$M = \mathfrak{a}_1 A_1 + \cdots + \mathfrak{a}_n A_n,$$

where $A_i \in K^l$ is the i-th row of A. We can construct a pseudo-basis from a pseudo-generating set by using the Hermite normal form (HNF) over Dedekind domains (see [2, Th. 1.4.6]). Note that this canonical form is also refered to as the pseudo-HNF in [2, 1.4]. In this paper we simply call it HNF, but we implicitly refer to the standard HNF over $\mathbb{Z}$ when dealing with an integer matrix. Assume A is of rank l (in particular $n \geq l$), then there exists an $n \times n$ matrix $U = (u_{i,j})$ and n non-zero ideals $\mathfrak{b}_1, \cdots, \mathfrak{b}_n$ satisfying

1. $\forall i, j, u_{i,j} \in \mathfrak{b}_i^{-1}\mathfrak{a}_j$.
2. $\mathfrak{a} = \det(U)\mathfrak{b}$ for $\mathfrak{a} = \prod_i \mathfrak{a}_i$ and $\mathfrak{b} = \prod_i \mathfrak{b}_i$.
3. The matrix UA is of the form

$$UA = \begin{pmatrix} 1 & 0 & \dots & 0 \\ \vdots & 1 & \ddots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ * & * & \dots & 1 \\ \hdashline & & (0) & \end{pmatrix}.$$

4. $M = \mathfrak{b}_1\omega_1 \oplus \cdots \oplus \mathfrak{b}_l\omega_l$ where $\omega_1, \cdots \omega_l$ are the first l rows of UA.

In general, the algorithm of [2] for computing the HNF of a pseudo-matrix takes exponential time, but as in the integer case, there exists a modular one which is polynomial in the dimensions of A, the degree of K, and the bit size of the modulus. Note that in the case of a pseudo matrix representing an $\mathcal{O}_K$-module M, the modulus is an integral multiple of the determinantal ideal $\mathfrak{g}(M)$, which is generated by all the ideals of the form

$$\det_{i_1, \cdots, i_l}(A) \cdot \mathfrak{a}_{i_1} \cdots \mathfrak{a}_{i_l},$$

where $\det_{i_1, \cdots, i_l}(A)$ is the determinant of the $l \times l$ minor consisting of the last l columns of rows of indices $i_1, \cdots, i_l$. The determinantal ideal is a rather involved structure, except in the case $l = n$. In applications, the modulus is frequently known. In the rest of the paper, we restrict ourselves to the case of an $n \times n$ matrix A of rank n. One can immediatly derive polynonmial time algorithms for the rectangular case, and for the case of a singular matrix A.

4. NOTION OF SIZE

To ensure that our algorithm for computing an HNF basis of an $\mathcal{O}_K$-module runs in polynomial time, we need a notion of size that bounds the bit size required to represent ideals and field elements. An ideal $I \subseteq \mathcal{O}_K$ is given by the matrix $M^I \in \mathbb{Z}^{d \times d}$ of its basis expressed in an integral basis $\omega_1, \cdots, \omega_d$ of $\mathcal{O}_K$. If the matrix is in Hermite Normal Form, the size required to store it is therefore bounded by $d^2 \max_{i,j}\left(\log(|M^I_{i,j}|)\right)$, where $\log(x)$ is the base 2 logarithm of x. In the meantime, every coefficient of M^I is bounded by $|\det(M^I)| = \mathcal{N}(I)$ (see [3, Prop. 4.7.4]). Thus, we define the size of an ideal as

$$S(I) := d^2 \log(\mathcal{N}(I)).$$

If $\mathfrak{a} = (1/k)I$ is a fractional ideal of K, where $I \subseteq \mathcal{O}_K$ and $k \in \mathbb{Z}_{>0}$ is minimal, then the natural generalization of the notion of size is

$$S(\mathfrak{a}) := \log(k) + S(I),$$

where $\log(k)$ is the base 2 logarithm of k. We also define the size of elements of K. If $x \in \mathcal{O}_K$ can be written as $x = \sum_{i \leq d} x_i \omega_i$, where $x_i \in \mathbb{Z}$, then we define its size by

$$S(x) := d \log(\max_i |x_i|).$$

It can be generalized to elements $y \in K$ by writing $y = x/k$ where $x \in \mathcal{O}_K$ and k is a minimal positive integer, and by setting

$$S(y) := \log(k) + S(x).$$

In the litterature, the size of elements of K is often expressed with $\|x\|$. These two notions are in fact related.

PROPOSITION 1. *Let $x \in \mathcal{O}_K$, the size of x and its T_2-norm satisfy*

$$\log(\|x\|) \leq \tilde{O}\left(\frac{S(x)}{d} + d^2 + \log|\Delta_K|\right)$$
$$S(x) \leq \tilde{O}\left(d\left(d + \log(\|x\|)\right)\right).$$

PROOF. In appendix □

So, for all $x \in \mathcal{O}_K$, $S(x) = O(\log(\|x\|))$, and $\log(\|x\|) = O(S(x))$, where the constants are polynomial in d and $\log|\Delta_K|$.

COROLLARY 1. *Let $x, y \in \mathcal{O}_K$, their size satisfies*

$$S(xy) \leq \tilde{O}\left(d^3 + d\log|\Delta_K| + S(x) + S(y)\right).$$

5. COST MODEL

We assume that the module M satifies $M \subseteq \mathcal{O}_K^n$ and that $\mathcal{O}_K$ is given by an LLL-reduced integral basis $\omega_1, \cdots, \omega_d$ such that $\omega_1 = 1$. The computation of such a basis can be done by using [1, Cor. 3] to produce a good integral basis for $\mathcal{O}_K$ and then reducing it with the LLL algorithm [7]. In this section, we evaluate the complexity of the basic operations performed during our algorithm. We rely on standard number theoretic algorithms. We multiply two integers of bit size h in time $\mathcal{M}(h) \leq O(h\log(h)\log(\log(h)))$ using Schönhage-Strassen algorithm, while the addition of such integers is in $O(h)$, their division has complexity bounded by $O(\mathcal{M}(h))$, and the Euclidiean algorihm that provides their GCD has complexity $O(\log(h)\mathcal{M}(h))$ (see [11]). In the following, we also refer to two standard linear algebra algorithms, namely the HNF computation over the integers due to Storjohann [15] in complexity $\left(nmr^{\omega-1}\log|A|\right)^{1+o(1)}$ and Dixon's p-adic algorithm for solving linear systems in

$$\left(n^\omega \log|A|\right)^{1+o(1)},$$

where $A \in \mathbb{Z}^{m\times n}$ has rank r and has its entries bounded by $|A|$, and where $3 \geq \omega \geq 2$ is the exponent of the complexity of matrix multiplication. We need to perform additions, multiplications and inversions of elements of K, as well as of fractional ideals. There is no reference on the complexity of these operations, although many implementations can be found. We adress this problem in the rest of this section. We use $\tilde{O}$ to denote the complexity were all the logarithmic factors are omitted.

Elements x of K are represented as quotients of an element of $\mathcal{O}_K$ and a positive denominator. We add them naively while their multiplication is done by using a precomputed table of the $\omega_i\omega_j$ for $i, j \leq d$.

PROPOSITION 2. *Let $\alpha, \beta \in K$ such that $S(\alpha), S(\beta) \leq B$, then the following holds:*

1. *$\alpha + \beta$ can be computed in $\tilde{O}(dB)$*
2. *$\alpha\beta$ can be computed in $\tilde{O}\left(d^2(B + d^3 + d\log|\Delta_K|)\right)$*
3. *$\frac{1}{\alpha}$ can be computed in $\tilde{O}\left(d^{\omega-1}(B + d^3 + d\log|\Delta_K|)\right)$,*

PROOF. Adding α and β is straightforward. Multiplying them is done by storing a precomputed multiplication table for the $\omega_i\omega_j$. Finally, inverting α boils down to solving a linear system in the coefficients of $\frac{1}{\alpha}$. More details are given in appendix. □

Ideals of $\mathcal{O}_K$ are given by their HNF representation with respect to the integral basis $\omega_1, \cdots, \omega_d$ of $\mathcal{O}_K$. It consists of the HNF of the matrix representing the d generators of their $\mathbb{Z}$ basis as rows. Operations on this matrix yield the addition, multiplication and inverse of an integral ideal. The corresponding operations on fractional ideals are trivialy deduced by taking care of the denominator.

PROPOSITION 3. *Let $\mathfrak{a}$ and $\mathfrak{b}$ be fractional ideals of K such that $S(\mathfrak{a}), S(\mathfrak{b}) \leq B$, then the following holds:*

1. *$\mathfrak{a} + \mathfrak{b}$ can be computed in $\tilde{O}(d^{\omega+1}B)$,*
2. *$\mathfrak{ab}$ can be computed in $\tilde{O}(d^3(d^4 + d^2\log|\Delta_K| + B))$,*
3. *$1/\mathfrak{a}$ can be computed in $\tilde{O}\left(d^{2\omega}(d^4 + d^2\log|\Delta_K| + B)\right)$.*

PROOF. The addition of integral ideals $\mathfrak{a}$ and $\mathfrak{b}$ given by their HNF matrix A and B is given by the HNF of $\left(\frac{A}{B}\right)$. To multiply them, one has to compute the HNF of the matrix whose d^2 rows represent $\gamma_i\delta_j$ where $(\gamma_i)_{i\leq d}$ is an integral basis for $\mathfrak{a}$ and $(\delta_i)_{i\leq d}$ is an integral basis for $\mathfrak{b}$. Finally, following the approach of [3, 4.8.4], inverting $\mathfrak{a}$ boils down to solving a $d^2 \times (d + d^2)$ linear system. More details are given in appendix. □

Note that the reason why the dependency in B in the complexity of the addition of fractional ideals is slightly more than in the complexity of the multiplication is the way we deal with the denominators. In the case of integral ideals, the addition would be in $\tilde{O}(d^{\omega-1}B)$. The last operation that needs to be performed during our HNF algorithm is the multiplication between an element of K and a fractional ideal.

PROPOSITION 4. *Let $\alpha \in K$, a fractional ideal $\mathfrak{a} \subseteq K$ and B_1, B_2 such that $S(\mathfrak{a}) \leq B_1$ and $S(\alpha) \leq B_2$, then $\alpha\mathfrak{a}$ can be computed in expected time bounded by*

$$\tilde{O}\left(d^\omega\left(d^3 + d\log|\Delta_K| + \frac{B_1}{d} + B_2\right)\right).$$

PROOF. If $\gamma_1, \cdots, \gamma_d$ is an integral basis for $\mathfrak{a} \subseteq \mathcal{O}_K$, then $(\alpha\gamma_i)_{i\leq d}$ is one for $(\alpha)\mathfrak{a}$. The HNF of the matrix representing these elements leads to the desired result. More details are given in appendix. □

6. THE NORMALIZATION

The normalization is the key difference between our approach and the one of Cohen [2, 1.5]. It is the strategy that prevents the coefficient swell by calculating a pseudo-basis for which the ideals are integral with size bounded by the field's invariants. Given a one-dimensional $\mathcal{O}_K$-module $\mathfrak{a}A \subseteq \mathcal{O}_K^n$ where $\mathfrak{a}$ is a fractional ideal of K, and $A \in K^n$, we find $b \in K$ such that the size taken to represent our module as $(b\mathfrak{a})(A/b)$ is reasonably bounded. Indeed, any non trivial module can be represented by elements of arbitrary large size, which would cause a significant slow-down in our algorithm.

The first step to our normalization is to make sure that $\mathfrak{a}$ is integral. This allows us to bound the denominator of the coefficients of the matrix when manipulating its rows during the HNF algorithm. If $k \in \mathbb{Z}$ is the denominator of $\mathfrak{a}$, then replacing $\mathfrak{a}$ by $k\mathfrak{a}$ and A by A/k increases the size needed to represent our module via the growth of all the denominators of the coefficients of $A \in K^n$. Thus, after this operation, the size of each coefficient a_i of A is bounded by $S(a_i) + S(\mathfrak{a})$.

We can now assume that our one-dimensional module is of the form $\mathfrak{a}A$ where $\mathfrak{a} \subseteq \mathcal{O}_K$ and $A \in K^n$ at the price of a slight growth of its size. The next step of normalization is to express our module as $\mathfrak{a}'A'$ where $A' \in K^n$ and $\mathfrak{a}' \subseteq \mathcal{O}_K$ such that $\mathcal{N}(\mathfrak{a}')$ only depends on invariants of the field. To do this, we invert $\mathfrak{a}$ and write it as

$$\mathfrak{a}^{-1} = \frac{1}{k}\mathfrak{b},$$

where $k \in \mathbb{Z}_{>0}$ and $\mathfrak{b} \subseteq \mathcal{O}_K$. As $\mathcal{N}(\mathfrak{a}) \in \mathfrak{a}$, we have $\mathcal{N}(\mathfrak{a})\mathfrak{a}^{-1} \subseteq \mathcal{O}_K$ and thus $k \leq \mathcal{N}(\mathfrak{a})$. Therefore,

$$\mathcal{N}(\mathfrak{b}) \leq \frac{\mathcal{N}(k)}{\mathcal{N}(\mathfrak{a})} \leq \frac{k^d}{\mathcal{N}(\mathfrak{a})} \leq \mathcal{N}(\mathfrak{a})^{d-1}.$$

Then we use the LLL algorithm to find an element $\alpha \in \mathfrak{b}$ such that

$$\|\alpha\| \leq d^{1/2} 2^{d/2} |\Delta_K|^{1/2d} \mathcal{N}(\mathfrak{b})^{1/d}.$$

Our reduced ideal is

$$\mathfrak{a}' := \left(\frac{\alpha}{k}\right)\mathfrak{a} \subseteq \mathfrak{a}^{-1}\mathfrak{a} = \mathcal{O}_K.$$

The integrality of $\mathfrak{a}'$ comes from the definition of $\mathfrak{b}^{-1}$ and the fact that $\alpha \in \mathfrak{b}$. From the arithmetic-geometric mean, we know that $\mathcal{N}(\alpha) \leq \frac{\|\alpha\|^d}{d^d}$, therefore

$$\mathcal{N}(\alpha) \leq 2^{d^2/2}\sqrt{|\Delta_K|}\mathcal{N}(\mathfrak{b}),$$

and the norm of the reduced ideal can be bounded by $\mathcal{N}(\mathfrak{a}') \leq 2^{d^2/2}\sqrt{|\Delta_K|}$. On the other hand, we set $A' := (k/\alpha)A$, which induces a growth of the coefficients a_i of A. Indeed, each a_i is multiplied by (k/α).

PROPOSITION 5. *The size of the normalized module $\mathfrak{a}'A'$ of $\mathfrak{a}A \subseteq K^n$ satisfies*

$$S(a_i') \leq \tilde{O}\left(d^3 + d\log|\Delta_K| + S(\mathfrak{a}) + S(a_i)\right)$$
$$S(\mathfrak{a}') \leq \tilde{O}\left(d^3 + d\log|\Delta_K|\right)$$

PROOF. From Corollary 1 we know that

$$S\left(\frac{a_i k}{\alpha}\right) \leq \tilde{O}\left(d^3 + d\log|\Delta_K| + \frac{S(\mathfrak{a})}{d} + S(a_i) + S\left(\frac{1}{\alpha}\right)\right)$$

In addition, if $\frac{1}{\alpha} = \frac{x}{k'}$ where $x \in \mathcal{O}_K$ and $k' \in \mathbb{Z}_{>0}$, then

$$S\left(\frac{1}{\alpha}\right) \leq \tilde{O}\left(\log(k') + d(d + \log\|x\|)\right).$$

On the one hand, we have

$$k' \leq \mathcal{N}(\alpha) \leq 2^{d^2/2}\sqrt{|\Delta_K|}\mathcal{N}(\mathfrak{a})^{d-1},$$

and on the other hand, we need to bound $\|x\|$. We notice that since $\mathcal{N}(\alpha) \in \mathbb{Q}$, $\forall j \leq d$, $\mathcal{N}(\alpha) = \alpha\beta = \sigma_j(\alpha\beta)$. We also know that $\forall j$, $|\sigma_j(\alpha)| \leq \|\alpha\|$. Therefore,

$$\forall j \leq d,\ |\sigma_j(x)| = \frac{\mathcal{N}(\alpha)}{|\sigma_j(\alpha)|} = \prod_{i \neq j} |\sigma_i(\alpha)| \leq \|\alpha\|^{d-1}.$$

Therefore $\|x\| \leq \sqrt{d}\|\alpha\|^{d-1}$, and thus

$$S\left(\frac{1}{\alpha}\right) \leq \tilde{O}\left(d^3 + d\log|\Delta_K| + S(\mathfrak{a})\right).$$

□

Our normalization, summarized in Algorithm 1, was performed at the price of a reasonable growth in the size of the object we manipulate. Let us now evaluate its complexity.

Algorithm 1 Normalization of a one-dimensional module

Input: $A \in K^n$, fractional ideal $\mathfrak{a}$ of K.
Output: $A' \in K^n$, $\mathfrak{a}' \subseteq \mathcal{O}_K$ such that $\mathcal{N}(\mathfrak{a}') \leq 2^{d^2/2}\sqrt{|\Delta_K|}$ and $\mathfrak{a}A = \mathfrak{a}'A'$.
1: $\mathfrak{a} \leftarrow k_0\mathfrak{a}$, $A \leftarrow A/k_0$ where k_0 is the denominator of $\mathfrak{a}$.
2: $\mathfrak{b} \leftarrow k\mathfrak{a}^{-1}$ where k is the denominator of $\mathfrak{a}^{-1}$.
3: Let α be the first element of an LLL-reduced basis of $\mathfrak{b}$.
4: $\mathfrak{a}' \leftarrow \left(\frac{\alpha}{k}\right)\mathfrak{a}$, $A' \leftarrow \left(\frac{k}{\alpha}\right)A$.
5: **return** $\mathfrak{a}'$, A'.

PROPOSITION 6. *Let B_1, B_2 such that $S(\mathfrak{a}) \leq B_1$ and $\forall i, S(a_i) \leq B_2$, then the complexity of Algorithm 1 is bounded by*

$$\tilde{O}\left(nd^2(d^3 + B_1 + B_2 + d\log|\Delta_K|)\right).$$

PROOF. The inversion of $\mathfrak{a}$ is performed in time

$$\tilde{O}\left(d^{2\omega}(d^4 + d^2\log|\Delta_K| + B_1)\right),$$

by using Proposition 3. Then, the LLL-reduction of the basis of $\mathfrak{b}$ is done by the L^2 algorithm of Stehlé and Nguyen [12] in expected time bounded by

$$\tilde{O}\left(d^3\left(d + \frac{S(\mathfrak{b})}{d^2}\right)\frac{S(\mathfrak{b})}{d^2}d\right) \leq \tilde{O}\left(d^2 S(\mathfrak{a})(d^2 + S(\mathfrak{a}))\right).$$

Then, computing $(\alpha/k)\mathfrak{a}$ is the multiplication of the ideal $\mathfrak{a}$ by the element α/k which satisfies

$$S(\alpha/k) \leq \tilde{O}\left(d^2 + \log|\Delta_K| + S(\mathfrak{a})/d\right).$$

This takes $\tilde{O}\left(d^{\omega-1}(S(\mathfrak{a}) + d^4 + d^2\log|\Delta_K|)\right)$. Finally, computing $k(1/\alpha)A$ consists of inverting α with $S(\alpha) \leq \tilde{O}(d^3 + \log|\Delta_K| + B_1/d)$, which takes

$$\tilde{O}\left(d^{\omega-1}(d^3 + B_1/d + d\log|\Delta_K|)\right),$$

and performing $n+1$ multiplications between elements of size bounded by $\tilde{O}(d^3 + B_1 + B_2 + d\log|\Delta_K|)$, which is done in time

$$\tilde{O}\left(nd^2(d^3 + B_1 + B_2 + d\log|\Delta_K|)\right).$$

The result follows from the combination of the above expected times and from the fact that $2 \leq \omega \leq 3$. □

7. REDUCTION MODULO A FRACTIONAL IDEAL

To achieve a polynomial complexity for our HNF algorithm, we reduce some elements of K modulo ideals whose norm can be reasonably bounded. We show in this section how to bound the norm of a reduced element with respect to the norm of the ideal and invariants of K. Let $\mathfrak{a}$ be a fractional ideal of K, and $x \in K$. Our goal is to find $\overline{x} \in K$ such that $\|\overline{x}\|$ is bounded, and that $x - \overline{x} \in \mathfrak{a}$.

The reduction algorithm consists of finding an LLL-reduced basis $r_1, \cdots, r_d$ of $\mathfrak{a}$ and to decompose

$$x = \sum_{i \leq d} x_i r_i.$$

Then, we define

$$\overline{x} := x - \sum_{i \leq d} \lfloor x_i \rceil r_i.$$

PROPOSITION 7. *Let $x \in K$ and $\mathfrak{a}$ be a fractional ideal of K, then Algorithm 2 returns $\overline{x}$ such that $x - \overline{x} \in \mathfrak{a}$ and*

$$\|\overline{x}\| \leq d^{3/2} 2^{d/2} \mathcal{N}(\mathfrak{a})^{1/d} \sqrt{|\Delta_K|}.$$

PROOF. In appendix □

Algorithm 2 Reduction modulo a fractional ideal

Input: $\alpha \in K$, fractional ideal $\mathfrak{a}$ of K.
Output: $\overline{\alpha} \in K$ such that $\alpha - \overline{\alpha} \in \mathfrak{a}$ and $\|\overline{\alpha}\| \leq d^{3/2} 2^{d/2} \mathcal{N}(\mathfrak{a})^{1/d} \sqrt{|\Delta_K|}$.

```
if ||α|| ≤ d^{3/2} 2^{d/2} N(a)^{1/d} sqrt(|Δ_K|) or α = 1 then
    return α.
else
    Compute an LLL-reduced basis (r_i)_{i≤d} of a.
    Decompose α = Σ_{i≤d} x_i r_i.
    ᾱ ← α − Σ_{i≤d} ⌊x_i⌉ r_i.
    return ᾱ.
end if
```

PROPOSITION 8. *Let B_1, B_2 such that $S(\mathfrak{a}) \leq B_1$ and $S(\alpha) \leq B_2$, then the complexity of Algorithm 2 is bounded by*

$$\tilde{O}\left(B_1(d^3 + B_1) + d^{\omega-1} B_2 + d^{\omega+2}\right)$$

PROOF. To compute the LLL-reduced basis of $\mathfrak{a}$, we LLL-reduce the integral ideal $k\mathfrak{a}$ where $k \in \mathbb{Z}_{>0}$ is the denominator of $\mathfrak{a}$. Then, we express x with respect to the basis of $k\mathfrak{a}$ where $x \in \mathcal{O}_K$ satifies $\alpha = x/a$ for $a \in \mathbb{Z}_{>0}$. Then we divide by the respective denominator at the extra cost of d multiplications.

Using the L^2 algorithm of Stehlé and Nguyen [12] yields the reduced basis of $k\mathfrak{a}$ in expected time bounded by

$$\tilde{O}\left(d^3 \left(d + \frac{S(\mathfrak{a})}{d^2}\right) \frac{S(\mathfrak{a})}{d^2} d\right) \leq \tilde{O}\left(S(\mathfrak{a})(d^3 + S(\mathfrak{a}))\right).$$

Then, expressing x with respect to the reduced basis of $k\mathfrak{a}$ costs

$$\tilde{O}\left(d^\omega \left(\frac{S(\mathfrak{a})}{d^2} + d \log |\Delta_K| + d^2 + \frac{S(x)}{d}\right)\right).$$

Finally, the subtraction and the division by the denominators are in

$$\tilde{O}\left(d \frac{S(\alpha)}{d}\right).$$

□

8. MODULAR HNF ALGORITHM

Let $M \subseteq \mathcal{O}_K^n$ be an $\mathcal{O}_K$-module. We use a variant of the modular version of [2, Alg. 1.4.7] which ensures that the current pseudo-basis $[\mathfrak{a}_i, A_i]_{i \leq n}$ of the module satisfies $\mathfrak{a} \subseteq \mathcal{O}_K$ at every step of the algorithm. This extra feature allows us to bound the denominator of coefficients of the matrix whose rows we manipulate. Algorithm 3 computes the HNF modulo the determinantal ideal $\mathfrak{g}$, and Algorithm 4 recovers an actual HNF for M. In this section, we discuss the differences between Algorithms 3 and 4 and their equivalent in [2, 1.4].

After the original normalization, all the ideals are integral. As $M \subseteq \mathcal{O}_K^n$, we immediatly deduce that the ideal $\mathfrak{d}$ created at Step 6 of Algorithm 3 is integral as well. In addition, from the definition of the inverse of an ideal we also have that

$$\frac{b_{i,i} \mathfrak{b}_i b_{i,j} \mathfrak{b}_j}{b_{i,j} \mathfrak{b}_j + b_{i,i} \mathfrak{b}_i} \subseteq \mathcal{O}_K,$$

which allows us to conclude that the update of $(\mathfrak{b}_i, \mathfrak{b}_j)$ performed at Step 9 of Algorithm 3 preserves the fact that our ideals are integral.

Algorithm 3 HNF of a full-rank square pseudo-matrix modulo $\mathfrak{g}$

Input: $A \in K^{n \times n}$, $\mathfrak{a}_1, \cdots, \mathfrak{a}_n$, $\mathfrak{g}$.
Output: pseudo-HNF B, $\mathfrak{b}_1, \cdots, \mathfrak{b}_n$ modulo $\mathfrak{g}$.

```
1: B ← A, b_i ← a_i, j ← n.
2: Normalize [(B_i), (b_i)]_{i≤n} with Algorithm 1
3: while j ≥ 1 do
4:    i ← j − 1.
5:    while i ≥ 1 do
6:       d ← b_{i,j} b_i + b_{j,j} b_j
7:       Find u ∈ b_i d^{-1} and v ∈ b_j d^{-1} such that b_{i,j} u +
         b_{j,j} v = 1 with [2, Th. 1.3.3].
8:       (B_i, B_j) ← (b_{j,j} B_i − b_{i,j} B_j, u B_i + v B_j).
9:       (b_i, b_j) ← (b_{i,j} b_i b_{j,j} b_j d^{-1}, d).
10:      Normalize b_i, B_i with Algorithm 1.
11:      Reduce B_i modulo g b_i^{-1} and B_j modulo g b_j^{-1} with
         Algorithm 2.
12:      i ← i − 1.
13:   end while
14:   j ← j − 1.
15: end while
16: return (b_i)_{i≤n}, B.
```

The normalization and reduction at Step 10-11 allow us to keep the size of the B_i and of the $\mathfrak{b}_i$ reasonably bounded by invariants of K and the dimension of the module. By doing so, we give away some information about the module M.

However, algorithm 4 allows us to recover M, as we state in Proposition 9.

PROPOSITION 9. *The $\mathcal{O}_K$-module defined by the pseudo-basis $[(W_i),(\mathfrak{c}_i)]$ obtained by applying Algorithm 4 to the HNF of M modulo $\mathfrak{g}(M)$ satisfies*

$$\mathfrak{c}_1 W_1 + \cdots + \mathfrak{c}_n W_n = M.$$

PROOF. The proof of this statement essentially follows its equivalent for matrices over the integers. It consists of showing that $W := \sum_i \mathfrak{c}_i$ and $M := \sum_i \mathfrak{a}_i A_i$ have the same determinantal ideal and that $W \subseteq A$, and then showing that this implies that $W = M$. A more complete proof is given in appendix. □

Algorithm 4 Eucledian reconstruction of the HNF

Input: $B \in K^{n\times n}$, $\mathfrak{b}_1, \cdots, \mathfrak{b}_n$ output of Algorithm 3 modulo $\mathfrak{g}$ for $M \subseteq \mathcal{O}_K^n$.
Output: An HNF W,$\mathfrak{c}_1, \cdots, \mathfrak{c}_n$ for M.
1: $j \leftarrow n$, $\mathfrak{g}_j \leftarrow \mathfrak{g}$.
2: **while** $j \geq 1$ **do**
3: $\quad \mathfrak{c}_j \leftarrow \mathfrak{b}_j + \mathfrak{g}_j$.
4: $\quad$ Find $u \in \mathfrak{b}_j\mathfrak{d}^{-1}$ and $v \in \mathfrak{g}\mathfrak{c}_j^{-1}$ such that $u + v = 1$.
5: $\quad W_j \leftarrow uB_j \bmod \mathfrak{g}\mathfrak{c}_j^{-1}$.
6: $\quad \mathfrak{g}_j \leftarrow \mathfrak{g}_j\mathfrak{c}_j^{-1}$.
7: $\quad j \leftarrow j - 1$.
8: **end while**
9: **return** $W, (\mathfrak{c}_i)_{i\leq n}$.

9. COMPLEXITY OF THE MODULAR HNF

Let us assume that we are able to compute the determinantal ideal $\mathfrak{g}$ of our module M in polynomial time with respect to the bit size of the invariants of the field and of $S(\mathfrak{g})$. We discuss the computation of $\mathfrak{g}$ in Section 10. In this section, we show that Algorithm 3 and Algorithm 4 are polynomial wih respect to the same parameters. This result is analogous to the case of integers matrices. Indeed, the only thing we need to verify is that the size of the elements remains reasonably bounded during the algorithm.

In Algorithm 3, the coefficient explosion is prevented by the modular reduction of Step 11. It ensures that

$$\forall i_1, i_2 < j,\ \|b_{i_1,i_2}\| \leq d^{3/2}2^{d/2}\mathcal{N}(\mathfrak{g}\mathfrak{b}_{i_1}^{-1})^{1/d}\sqrt{|\Delta_K|}.$$

This is not enough to prevent the explosion since b_{i_1,i_2} might not be integral. Therefore, there is a minimal $k \in \mathbb{Z}_{>0}$ such that $kb_{i_1,i_2} \in \mathcal{O}_K$, which we need to bound to ensure that $S(b_{i_1,i_2})$ remains bounded as well. We know that $b_{i,j}\mathfrak{b}_i \subseteq \mathcal{O}_K$, and that $\mathfrak{b}_i$ is integral. Thus, $\mathcal{N}(k) \mid \mathcal{N}(\mathfrak{b}_{i_1})$, which in turns implies that $k \leq \mathcal{N}(\mathfrak{b}_{i_1})$. As on the other hand, the normalization of Step 10 ensures that $\mathcal{N}(\mathfrak{b}_{i_1}) \leq 2^{d^2/2}\sqrt{|\Delta_K|}$, we conclude that after Step 11,

$$S(b_{i_1,i_2}) \leq \tilde{O}\left(d^2 + d\log|\Delta_K| + \frac{S(\mathfrak{g})}{d^2}\right).$$

In Algorithm 3, we last manipulate B_j and $\mathfrak{b}_j$ when the index j is the pivot. In that case, we cannot use the normalization to bound the size since we require that $b_{j,j} = 1$. However we reduce B_j modulo $\mathfrak{g}\mathfrak{b}_j$, which means that

$$\forall i \leq j,\ \|b_{i,j}\| \leq d^{3/2}2^{d/2}\mathcal{N}(\mathfrak{g}\mathfrak{b}_i^{-1})^{1/d}\sqrt{|\Delta_K|}.$$

In addition, the arithmetic-geometric tells us that $\|b_{j,j}\| \geq \sqrt{d}\mathcal{N}(b_{i,j})^{1/d}$, which in turn implies that

$$\forall i \leq j,\ \mathcal{N}(b_{i,j}\mathfrak{b}_i) \leq d^d 2^{d^2/2}\mathcal{N}(\mathfrak{g})^d|\Delta_K|^{d/2}. \tag{1}$$

As we know that

$$\mathcal{N}(b_{i,j}\mathfrak{b}_i + b_{j,j}\mathfrak{b}_j) \leq \max\left(\mathcal{N}(b_{i,j}\mathfrak{b}_i), \mathcal{N}(b_{j,j}\mathfrak{b}_j)\right),$$

we therefore know that after Step 9

$$\mathcal{N}(\mathfrak{b}_j) \leq d^d 2^{d^2/2}\mathcal{N}(\mathfrak{g})^d|\Delta_K|^{d/2},$$

which allows us to bound the size of the denominators in the j-th row the same way we did for the rows of index $i_1 < j$:

$$\forall i \leq j,\ S(b_{i,j}) \leq \tilde{O}\left(d^2 + d\log|\Delta_K| + \frac{S(\mathfrak{g})}{d^2}\right).$$

PROPOSITION 10. *The complexity of Algorithm 3 is in*

$$\tilde{O}\left(n^3 d^2\left(d^3 + d^2\log|\Delta_K| + S(\mathfrak{g})\right)^2\right).$$

PROOF. Steps 6 to 11 of Algorithm 3 are repeated $O(n^2)$ times. Let us analyze their complexity. First, at Step 6 we have

$$S(b_{i,j}) \leq \tilde{O}\left(d^3 + \log|\Delta_K| + \frac{S(\mathfrak{g})}{d^2}\right) \tag{2}$$

$$S(\mathfrak{b}_i) \leq \tilde{O}(d^3 + \log|\Delta_K|) \tag{3}$$

so from Proposition 4, computing $b_{i,j}\mathfrak{b}_i$ takes

$$\tilde{O}\left(d^{\omega-2}(d^5 + d^3\log|\Delta_K| + S(\mathfrak{g}))\right).$$

Then, from Proposition 3 and (1),

$$S(b_{i,j}\mathfrak{b}_i) \leq \tilde{O}\left(d^4 + dS(\mathfrak{g}) + d^3\log|\Delta_K|\right),$$

and computing $\mathfrak{d}$ costs

$$\tilde{O}\left(d^{\omega+2}(d^3 + d^2\log|\Delta_K| + S(\mathfrak{g}))\right).$$

As $S(\mathfrak{d}) \leq S(b_{i,j}\mathfrak{b}_i)$, computing $\mathfrak{d}^{-1}$ takes

$$\tilde{O}\left(d^{2\omega+1}\left(d^3 + d^2\log|\Delta_K| + S(\mathfrak{g})\right)\right).$$

From [3, 4.8.4], this is done by solving a linear system on a matrix D satisfying

$$\log|D| \leq \tilde{O}\left(d^2 + \log|\Delta_K| + \frac{S(\mathfrak{g})}{d^2}\right),$$

and the coefficients of the HNF matrix of $\mathfrak{d}$ are those of a matrix M satisfying $\log|\det(M)| \leq \tilde{O}\left(d^2\log|D|\right)$. Therefore, we have

$$S(\mathfrak{d}^{-1}) \leq d^2\log|\det(M)| \leq \tilde{O}\left(d^2\left(d^4 + d^2\log|\Delta_K| + S(\mathfrak{g})\right)\right).$$

As $S(\mathfrak{b}_i), S(\mathfrak{b}_j) \leq \tilde{O}(d^3 + \log|\Delta_K|)$, computing $\mathfrak{b}_i\mathfrak{d}^{-1}$ and $\mathfrak{b}_j\mathfrak{d}^{-1}$ takes

$$\tilde{O}\left(d^5\left(d^4 + d^2\log|\Delta_K| + S(\mathfrak{g})\right)\right).$$

Then, from [2, Th. 1.3.3], computing u and v is done by finding $u' \in b_{i,j}\mathfrak{b}_i\mathfrak{d}^{-1}$ and $v' \in b_{j,j}\mathfrak{b}_j\mathfrak{d}^{-1}$ such that $u' + v' =$

1 and returning $u := u'/b_{i,j}$ and $v := v'/b_{j,j}$. Let $I_i := b_{i,j}\mathfrak{b}_i\mathfrak{d}^{-1} \subseteq \mathcal{O}_K$. Then, from [2, Prop. 1.3.1] computing u', v' is done at the cost of an HNF computation of a $2d \times d$ matrix whose entries have their size bounded by $\log(\mathcal{N}(I_j))$. This cost is in

$$\tilde{O}\left(d^{\omega}(d^3 + d^2 \log|\Delta_K| + S(\mathfrak{g}))\right).$$

In addition, $S(u'), S(v') \leq \tilde{O}(d^4 + d^3 \log|\Delta_K| + dS(\mathfrak{g}))$. Then, by using the same methdods as in the proof of Proposition 5, we know that $S\left(\frac{1}{b_{i,j}}\right) \leq \tilde{O}\left(d^3 + \frac{S(\mathfrak{g})}{d} + d^2 \log|\Delta_K|\right)$ while Proposition 2 ensures us that inverting $b_{i,j}$ is done in

$$\tilde{O}\left(d^{\omega-1}\left(d^3 + d\log|\Delta_K| + \frac{S(\mathfrak{g})}{d^2}\right)\right).$$

Then, calculating $u'/b_{i,j}$ and $v'/b_{j,j}$ is done in time bounded by

$$\tilde{O}\left(d^2\left(d^4 + d^3\log|\Delta| + dS(\mathfrak{g})\right)\right),$$

and by Corollary 1, we know that

$$S(u), S(v) \leq \tilde{O}\left(d^4 + d^3 \log|\Delta_K| + dS(\mathfrak{g})\right).$$

Then, from Proposition 2 and (2), the expected time for Step 8 is bounded by

$$\tilde{O}\left(nd^2(d^4 + d^3\log|\Delta_K| + dS(\mathfrak{g}))\right).$$

In addition, after Step 8, we have

$$S(b_{i,j}) \leq \tilde{O}\left(d^4 + d^3\log|\Delta_K| + dS(\mathfrak{g})\right).$$

Then, from Proposition 3 and the bounds on $S(b_{i,j}\mathfrak{b}_i)$ and $S(\mathfrak{d}^{-1})$ computed above, Step 9 takes

$$\tilde{O}\left(d^5\left(d^4 + d^2\log|\Delta_K| + S(\mathfrak{g})\right)\right).$$

By using Proposition 6, we bound the time taken by Step 10 by

$$\tilde{O}\left(nd^3(d^3 + d^2\log|\Delta_K| + S(\mathfrak{g}))\right),$$

Finally, from the bound on $S(b_{i,j})$ after Step 8 and Proposition 8, Step 11 takes

$$\tilde{O}\left(nd^2\left(d^3 + d^2\log|\Delta_K| + S(\mathfrak{g})\right)^2\right).$$

□

The Euclidian reconstruction of Algorithm 4 can be seen as another pivot operation between the two one-dimensional $\mathcal{O}_K$-modules $\mathfrak{b}_j B_j$ and $\mathfrak{g}_j e_j$ for each $j \leq n$. We can therefore bound the entries of W by the same method as for Step 6-11 of Algorithm 3, we the extra observation

$$\mathcal{N}(\mathfrak{g}_j) \leq \mathcal{N}(\mathfrak{g}).$$

Therefore, we showed that we could bound the size of the objects that are manipulated throughout the algorithm by values that are polynomial in terms of n, d, $S(\mathfrak{g})$ and $\log(|\Delta_K|)$, and that the complexity of the HNF algorithm was polynomial in these parameters.

10. COMPUTING THE MODULUS

Let us assume that $A \in \mathcal{O}_K^{n\times n}$. If it is not the case, then we need to multiply by the common denominator k of the entries of A and return $\det(kA)/k^n$. In this section, we describe how to compute $\mathfrak{g}$ in polynomial time with respect to n, d, $\log|\Delta_K|$ and the size of the entries of A. The idea is to compute $\det(A) \mod (p)$ for a sufficiently large prime number p. In practice, one might prefer to compute $\det(A) \mod (p_i)$ for several prime numbers $p_1, \cdots, p_l$ and recombine the values via the chinese remainder theorem, but for the sake of simplicity, we only describe that procedure for a single prime. Once $\det(A)$ is computed in polynomial time, we return

$$\mathfrak{g} = \det(A) \cdot \mathfrak{a}_1 \cdots \mathfrak{a}_n.$$

The first step consists of evaluating how large p should be to ensure that we recover $\det(A)$ uniquely. As $p\omega_1, \cdots, p\omega_d$ is an integral basis for (p), it suffices that $p \geq \max_i |a_i|$ where $\det(A) = \sum_i a_i\omega_i$. As $\max_i |a_i| \leq 2^{3d/2}\|\det(A)\|$, it suffices to bound $\|\det(A)\|$. We first compute an upper bound on $|\sigma(\det(A))|$ for the d complex embeddings σ of K via Hadamard's inequality and then we deduce a bound on $\|\det(A)\|$. Let $\sigma : K \to \mathcal{C}$, we know from Hadamard's inequality that

$$|\sigma(\det(A))| \leq B^n n^{n/2},$$

where B is a bound on $\sigma(a_{i,j})$. Such a bound can be derived from the size of the coefficient of A by using

$$\forall x,\ \forall i\ |\sigma_i(x)| \leq \left(\max_j |x_j|\right) d^{3/2} 2^{d^2/2}\sqrt{|\Delta_K|}.$$

This way, we see that $B := 2^{\max_{i,j}\left(S(a_{i,j})\right)} d^{3/2} 2^{d^2/2}\sqrt{|\Delta_K|}$ suffices. Then, our bound on $\|\det(A)\|$ is simply

$$\|\det(A)\| \leq \sqrt{n} 2^{\max_{i,j}\left(S(a_{i,j})\right)} d^{3/2} 2^{d^2/2}\sqrt{|\Delta_K|}.$$

Algorithm 5 Computation of $\det(A)$

Input: $A \in \mathcal{O}_K^{n\times n}$, $B > \max_{i,j}(S(a_{i,j}))$
Output: $\det(A)$.

1: Let $p \geq \sqrt{n}2^B d^{3/2} 2^{d^2/2}\sqrt{|\Delta_K|}$ be a prime.
2: **for** $\mathfrak{p}_i \mid (p)$ **do**
3: Compute $\det(A) \mod \mathfrak{p}_i$.
4: **end for**
5: Recover $\det(A) \mod (p)$ via successive applications of Algorithm 6
6: **return** $\det(A)$.

To reconstruct $\det(A) \mod (p)$ from $\det(A) \mod \mathfrak{a}_i$ for $i \leq d$, let us consider the simpler case of the reconstruction modulo two coprime ideals $\mathfrak{a}, \mathfrak{b}$ of $\mathcal{O}_K$. Let $M_\mathfrak{a}$ and $M_\mathfrak{b}$ be the matrices representing the $\mathbb{Z}$ basis of $\mathfrak{a}$ and $\mathfrak{b}$ in the integral basis $(\omega_i)_{i\leq d}$ of $\mathcal{O}_K$, and let $x, y, w \in \mathcal{O}_K$ such that

$$x = y \mod \mathfrak{a}$$
$$x = w \mod \mathfrak{b}.$$

We wish to compute $z \in \mathcal{O}_K$ such that $x = z \mod \mathfrak{a}\mathfrak{b}$. As in [2, Prop. 1.3.1], we can derive $a \in \mathfrak{a}, b \in \mathfrak{b}$ such that $a + b = 1$ from the HNF of $\left(\frac{M_\mathfrak{a}}{M_\mathfrak{b}}\right)$. Then, a solution to our CRT recomposition is given by

$$z := wa + yb.$$

Algorithm 6 CRT recomposition

Input: $\mathfrak{a}, \mathfrak{b} \subseteq \mathcal{O}_K$, $x, y, w \in \mathcal{O}_K$ such that $x = y \bmod \mathfrak{a}$ and $x = w \bmod \mathfrak{b}$.

Output: $z \in \mathcal{O}_K$ such that $x = z \bmod \mathfrak{a}\mathfrak{b}$.

1: Compute $a \in \mathfrak{a}, b \in \mathfrak{b}$ such that $a + b = 1$.
2: **return** z.

PROPOSITION 11. *Let $B > \max_{i,j}(S(a_{i,j}))$, then the complexity of Algorithm 5 is bounded by*

$$\tilde{O}\left(n^3 d^7 (d^2 + B + \log|\Delta_K|)^2\right).$$

PROOF. For each $\mathfrak{p}_i$, the computation of $\det(A) \bmod \mathfrak{p}_i$ consists of n^3 multiplications of reduced elements modulo $\mathfrak{p}_i$ followed by a reduction modulo $\mathfrak{p}_i$. Given our choice of p, we have

$$\log \mathcal{N}(\mathfrak{p}_i) \leq \tilde{O}\left(d(B + d^2 + \log|\Delta_K|)\right).$$

Therefore, the size of the elements $x \in \mathcal{O}_K$ involved in these multiplications satisfies

$$S(x) \leq \tilde{O}\left(d^2(d^2 + \log|\Delta_K| + B)\right).$$

The cost of the multiplications is in

$$\tilde{O}\left(d^4(d^2 + B + \log|\Delta_K|)\right),$$

while the mdular reductions cost

$$\tilde{O}\left(d^6(d^2 + B + \log|\Delta_K|)^2\right).$$

The time to reconstruct $\det(A) \bmod (p)$ corresponds to the computation of n^2 Hermite forms of $d^2 \times d$ integer matrices M such that $\log|M| \leq \log(\mathcal{N}(\mathfrak{p}_i))$. This takes

$$\tilde{O}\left(n^2 d^{\omega+3}(d^2 + B + \log|\Delta_K|)^2\right).$$

□

11. CONCLUSION

We described a polynomial time algorithm for computing the HNF basis of an $\mathcal{O}_K$-module. Our strategy relies on the one of Cohen [2, 1.4] who had conjectured that his modular algorithm was polynomial. The crucial difference between our algorithm and the one of [2, 1.4] is the normalization which allows us to prove the complexity to be polynomial. Without it, we cannot bound the denominator of the coefficients of the matrix when we recombine rows, even if they are reduced modulo the determinantal ideal. We provided a rigorous proof of the complexity of our method with respect to the dimension of the module, the size of the input and the invariants of the field. Our algorithm is the first polynomial time method for computing the HNF of an $\mathcal{O}_K$-module. This result is significant since other applications rely on the possibility of computing the HNF of an $\mathcal{O}_K$-module in polynomial time. In particular, Fieker and Stehlé [5] made this assumption in the analysis of their LLL algorithms for $\mathcal{O}_K$-modules. Our result has natural ramifications in cryptography through the LLL algorithm of Fieker and Stehlé [5], but it can also be used for list-decoding number field codes.

12. REFERENCES

[1] J. Buchmann and O. van Sprang. On short representations of orders and number fields, 1992. http://www.cdc.informatik.tu-darmstadt.de/reports/reports/short_rep_ord.ps.gz.

[2] H. Cohen. *Advanced topics in computational algebraic number theory*, volume 193 of *Graduate Texts in Mathematics*. Springer-Verlag, 1991.

[3] H. Cohen. *A course in computational algebraic number theory*, volume 138 of *Graduate Texts in Mathematics*. Springer-Verlag, 1991.

[4] C. Fieker. Minimizing representations over number fields. *J. Symb. Comput.*, 38(1):833–842, 2004.

[5] C. Fieker and D. Stehlé. Short bases of lattices over number fields. In G. Hanrot, F. Morain, and E. Thomé, editors, *Algorithmic Number Theory, 9th International Symposium, ANTS-IX, Nancy, France, July 19-23, 2010. Proceedings*, volume 6197 of *Lecture Notes in Computer Science*, pages 157–173. Springer, 2010.

[6] V. Guruswami. Constructions of codes from number fields. *IEEE Transactions on Information Theory*, 49(3):594–603, 2003.

[7] A. K. Lenstra, J. H. W. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, 1982.

[8] V. Lyubashevsky and D. Micciancio. Generalized compact knapsacks are collision resistant. In I. Wegener, V. Sassone, and B. Preneel, editors, *Proceedings of the 33rd international colloquium on automata, languages and programming - ICALP 2006*, volume 4052 of *Lecture Notes in Computer Science*, pages 144–155, Venice, Italy, July 2006. Springer-Verlag.

[9] D. Micciancio. Generalized compact knapsaks, cyclic lattices, and efficient one-way functions from worst-case complexity assumptions. In *Proceedings of the 43rd Annual Symposium on Foundations of Computer Science - FOCS 2002.*, pages 356–365, Vancouver, Canada, Nov. 2002. IEEE. Full version in Computational Complexity 16:365-411.

[10] D. Micciancio. Generalized compact knapsaks, cyclic lattices, and efficient one-way functions. *Computational Complexity*, 16(4):365–411, Dec. 2007. Prelim. in FOCS 2002.

[11] N. Möller. On schÃűnhage's algorithm and subquadratic integer gcd computation. *Mathematics of Computation*, 77:589–607, 2008.

[12] P. Q. Nguyen and D. Stehlé. An lll algorithm with quadratic complexity. *SIAM Journal on Computing*, 39(3):874–903, 2009.

[13] C. Peikert and A. Rosen. Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices. In *Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006*, pages 145–166, 2006.

[14] D. Stehlé, R. Steinfeld, K. Tanaka, and K. Xagawa. Efficient public-key encryption based on ideal lattices (extended abstract). In *Advances in Cryptology - ASIACRYPT 2008, 14th International Conference on the Theory and Application of Cryptology and Information Security, Melbourne, Australia, December 7-11, 2008. Proceedings*, volume 5912 of *LNCS*, pages 617–635. Springer, 2009.

[15] A. Storjohann. *Algorithms for Matrix Canonical Forms*. PhD thesis, Department of Computer Science, Swiss Federal Institute of Technology – ETH, 2000.

APPENDIX

A. DETAILED PROOFS OF STATEMENTS

A.1 notion of size

PROPOSITION 1. *Let $x \in \mathcal{O}_K$, the size of x and its T_2-norm satisfy*

$$\log(\|x\|) \leq \tilde{O}\left(\frac{S(x)}{d} + d^2 + \log|\Delta_K|\right)$$
$$S(x) \leq \tilde{O}\left(d\left(d + \log(\|x\|)\right)\right).$$

PROOF. Let us show how $S(x)$ and $\log(\|x\|)$ are related. First, we can assume [5, Lem. 1] that we choose an LLL-reduced integral basis $\omega_1, \cdots, \omega_d$ of $\mathcal{O}_K$ satisfying

$$\max_i \|\omega_i\| \leq \sqrt{d} 2^{d^2/2} \sqrt{|\Delta_K|}.$$

Then, we have

$$\begin{aligned}\forall i \leq d, |x|_i = |\sigma_i(x)| &= \left|\sum_{j\leq d} |x_j| \sigma_i(\omega_j)\right| \\ &\leq d\left(\max_i |x_i|\right) \|\omega_j\| \\ &\leq \left(\max_i |x_i|\right) d^{3/2} 2^{d^2/2} \sqrt{|\Delta_K|}.\end{aligned}$$

Therefore, $\log(\|x\|) \leq S(x) + d\log\left(d^{3/2}2^{d^2/2}\sqrt{|\Delta_K|}\right)$. On the other hand, we know from [5, Lem. 2] that for our choice of an integral basis of $\mathcal{O}_K$, we have

$$\forall x \in \mathcal{O}_K,\ S(x) \leq d\log\left(2^{3d/2}\|x\|\right).$$

□

A.2 Cost model

PROPOSITION 2. *Let $\alpha, \beta \in K$ such that $S(\alpha), S(\beta) \leq B$, then the following holds:*

1. *$\alpha + \beta$ can be computed in $\tilde{O}(dB)$*
2. *$\alpha\beta$ can be computed in $\tilde{O}\left(d^2(B + d^3 + d\log|\Delta_K|)\right)$*
3. *$\frac{1}{\alpha}$ can be computed in $\tilde{O}\left(d^{\omega-1}(B + d^3 + d\log|\Delta_K|)\right)$,*

where $\tilde{O}$ denotes the complexity whithout the logarithmic factors.

PROOF. Let $x, y \in \mathcal{O}_K$ and $a, b \in \mathbb{Z}_{>0}$ such that $\alpha = x/a$ and $\beta = y/b$. The first step of computing $\alpha + \beta$ consists of reducing them to the same denominator. This takes a time bounded by $\tilde{O}(dB)$. Then the addition of the numerators takes $\tilde{O}(dB)$, as well as and the simplification by the GCD of the denominator and the d coefficients.

For $i, j, k \leq d$, let $a_{i,j}^{(k)}$ be such that $\omega_i\omega_j = \sum_{k\leq d} a_{i,j}^{(k)}\omega_k$. From [5, Lem. 1], we know that $\forall i, \|\omega_i\| \leq \sqrt{d}2^{d^2/2}\sqrt{|\Delta_K|}$, and thus

$$\forall i, j \|\omega_i\omega_j\| \leq \|\omega_i\|\|\omega_j\| \leq d2^{d^2}|\Delta_K|.$$

Therefore, from Proposition 1, we have $\forall i, j, k, \log\left(|a_{i,j}^{(k)}|\right) \leq \tilde{O}(d^2 + \log|\Delta_K|)$. Then, if $x = \sum_{i\leq d} b_i\omega_i$ and $y = \sum_{j\leq d} c_j\omega_j$, we first need to compute b_ic_j for every $i, j \leq d$, which takes time $d^2\mathcal{M}(B/d)$. Then, we compute $(b_ic_j)a_{i,j}^{(k)}$ for $i, j, k \leq d$, which takes $\tilde{O}(d^3\mathcal{M}(2B/d + d^2 + \log|\Delta_K|)$. Then for each $k \leq d$, we compute $\sum_{i,j} b_ic_ja_{i,j}^{(k)}$, which is in $\tilde{O}(d(B/d + d^2 + \log|\Delta_K|))$. Finally, the multiplication of the denominators is in $\mathcal{M}(B)$, and the simplification of the numerator and denominator takes $\tilde{O}(d\mathcal{M}(B/d + d^2 + \log|\Delta_K|))$.

To invert $x = \sum_i b_i\omega_i$, we first define $A := (d_{j,k})_{j,k\leq d}$ by $d_{j,k} := \sum_i b_i a_{i,j}^{(k)}$, and notice that

$$\forall i,\ x\omega_i = \sum_i b_i\left(\sum_{k\leq d} a_{i,j}^{(k)}\omega_k\right) = \sum_{k\leq d} d_{j,k}\omega_k.$$

Inverting x boils down to finding $x_1, \cdots, x_d \in \mathbb{Q}$ such that $\sum_i xx_i\omega_i = 1$. It can be achieved by solving

$$XA = (1, 0, \cdots, 0).$$

We derive the complexity of this step by noticing that $\log|A| \leq 2^{B/d+d^2+3d/2}d|\Delta_K|$. From Hadamard's inequality, we know that the numerator and the denominator of x_i are bounded by

$$d^{d/2}|A|^d \leq 2^{d^3+3d^2/2+B}d^{3d/2}|\Delta_K|^d.$$

Multiplying all numerators by a where $\alpha = x/a$ costs

$$\tilde{O}\left(d\mathcal{M}(d^3 + B + d\log(|\Delta_K|)\right),$$

while reducing the ax_i to the same denominator and simplifying the expression can be done in

$$\tilde{O}\left(d(d^3 + B + d\log(|\Delta_K|))\right).$$

As $\omega \geq 2$, the complexity of the inversion is in fact dominated by the resolution of the linear system. □

PROPOSITION 3. *Let $\mathfrak{a}$ and $\mathfrak{b}$ be fractional ideals of K such that $S(\mathfrak{a}), S(\mathfrak{b}) \leq B$, then the following holds:*

1. *$\mathfrak{a} + \mathfrak{b}$ can be computed in $\tilde{O}(d^{\omega+1}B)$,*
2. *$\mathfrak{ab}$ can be computed in $\tilde{O}(d^3(d^4 + d^2\log|\Delta_K| + B))$,*
3. *$1/\mathfrak{a}$ can be computed in $\tilde{O}\left(d^{2\omega}(d^4 + d^2\log|\Delta_K| + B)\right)$.*

PROOF. Let $A, C \in \mathbb{Z}^{d\times d}$ in HNF form and $a, c \in \mathbb{Z}_{>0}$ such that $\mathfrak{a} = \frac{1}{a}\left(\sum_{i\leq d}\mathbb{Z}A_i\right)$ and $\mathfrak{b} = \frac{1}{c}\left(\sum_{i\leq d}\mathbb{Z}C_i\right)$, where A_i denotes the i-th row of A. Adding $\mathfrak{a}$ and $\mathfrak{b}$ is done by computing the HNF of $\left(\frac{cA}{aC}\right)$ and reducing the denominator. The complexity is bounded by the one of the HNF which is in $\tilde{O}(d^{\omega+1}B)$ since $\log|cA|, \log|aC| \leq B + B/d^2$.

Let $\gamma_1, \cdots, \gamma_d$ and $\delta_1, \cdots, \delta_d$ be integral elements such that

$$\mathfrak{a} = \frac{1}{a}\left(\mathbb{Z}\gamma_1 + \cdots + \mathbb{Z}\gamma_d\right)$$
$$\mathfrak{b} = \frac{1}{b}\left(\mathbb{Z}\delta_1 + \cdots + \mathbb{Z}\delta_d\right)$$

for $a, b \in \mathbb{Z}_{>0}$. We first compute $\gamma_i\delta_j$, which takes

$$\tilde{O}\left(d^3(S(\mathfrak{a}) + d^4 + d^2\log|\Delta_K|)\right).$$

Their size satisfies $S(\gamma_i\gamma_j) \leq \tilde{O}\left(d^3 + d\log|\Delta_K| + \frac{S(\mathfrak{a})}{d}\right)$. Then, we compute the HNF basis of the $\mathbb{Z}$-module generated by the $\gamma_i\delta_j$, which costs

$$\tilde{O}\left(d^\omega(d^4 + d^2\log|\Delta_K| + S(\mathfrak{a}))\right),$$

and we finally perform d^2 gcd reduction involving the product of the denominators which is bounded by $\tilde{O}(B)$.

Finally, we know from [3, 4.8.4] that finding the inverse of $\mathfrak{a}$ consists of calculating a basis of the nullspace of a matrix $D \in \mathbb{Z}^{(d^2+d)\times d^2}$ satisfying $\log|D| \leq \tilde{O}(d^2 + \log|\Delta_K| + B/d^2)$, and returning the HNF of its left $d\times d$ minor U. By using [15, Prop. 6.6], we find such a nullspace $M \in \mathbb{Z}^{d\times d^2}$ satisfying $|M| \leq d(\sqrt{d}|D|)^{2d}$ in expected time bounded by

$$\tilde{O}\left(d^{2+2\omega}\log|D|\right) \leq \tilde{O}\left(d^{2\omega}(d^4 + d^2\log|\Delta_K| + B)\right).$$

The HNF of U has complexity bounded by $\tilde{O}(d^{\omega+1}\log|M|) \leq \tilde{O}(d^{2+\omega}\log|D|)$. □

PROPOSITION 4. *Let $\alpha \in K$, a fractional ideal $\mathfrak{a} \subseteq K$ and B_1, B_2 such that $S(\mathfrak{a}) \leq B_1$ and $S(\alpha) \leq B_2$, then $\alpha\mathfrak{a}$ can be computed in expected time bounded by*

$$\tilde{O}\left(d^\omega\left(d^3 + d\log|\Delta_K| + \frac{B_1}{d} + B_2\right)\right).$$

PROOF. Let $x \in \mathcal{O}_K$ and $a \in \mathbb{Z}_{>0}$ such that $\alpha = x/a$ and let $k \in \mathbb{Z}_{>0}$ and $\gamma_1, \cdots, \gamma_d$ be an HNF basis for $\mathfrak{a}$. Then, $(x\gamma_i)_{i\leq d}$ is a $\mathbb{Z}$-basis for $(x)\mathfrak{a}$. We perform d multiplications $x\gamma_i$ where $S(\gamma_i) \leq B_1/d$ and $S(x) \leq B_2$. This costs

$$\tilde{O}\left(d^3\left(\frac{B_1}{d} + B_2 + d^3 + d\log|\Delta_K|\right)\right).$$

Then, from Corollary 1, we know that

$$S(x\gamma_i) \leq \tilde{O}\left(d^3 + d\log|\Delta_K| + S(x) + S(\gamma_i)\right).$$

Therefore, computing the HNF of the resulting matrix of entries bounded by $S(x\gamma_i)/d$ takes

$$\tilde{O}\left(S(x\gamma_i)d^\omega\right) \leq \tilde{O}\left(d^\omega\left(d^3 + d\log|\Delta_K| + S(x) + S(\gamma_i)\right)\right).$$

Finally, we multiply the denominators and reduce them by successive GCD computations in time $\tilde{O}(dS(x\gamma_i))$. □

A.3 Reduction modulo a fractional ideal

PROPOSITION 7. *Let $x \in K$ and $\mathfrak{a}$ be a fractional ideal of K, then Algorithm 2 returns $\overline{x}$ such that $x - \overline{x} \in \mathfrak{a}$ and*

$$\|\overline{x}\| \leq d^{3/2}2^{d/2}\mathcal{N}(\mathfrak{a})^{1/d}\sqrt{|\Delta_K|}.$$

PROOF. The LLL [7] algorithm allows us to compute a basis $(r_j)_{j\leq d}$ for I that satisfies

$$\|r_j\| \leq 2^{d/2}\sqrt{d}\mathcal{N}(I)^{1/d}\sqrt{|\Delta_K|}.$$

The same holds for a fractional ideal $\mathfrak{a}$ of K by multiplying the above relation by the denominator of $\mathfrak{a}$. Then, as $\lfloor x_j \rceil r_j \leq 1$, we see that

$$\|\overline{x}\| \leq d\max_j\|r_j\| \leq d^{3/2}2^{d/2}\mathcal{N}(\mathfrak{a})^{1/d}\sqrt{|\Delta_K|}.$$

□

A.4 The HNF

At the end of Algorithm 3, we obtain a pseudo-basis $[(B_i)_{i\leq n}, (\mathfrak{b}_i)_{i\leq n}]$ such that

$$\forall i \leq n\ \mathfrak{b}_iB_i \subseteq M + \mathfrak{g}e_i,$$

where $e_i := (0, 0, \cdots, 1, 0, \cdots, 0)$ is the i-th vector of the canonical basis of K^n. However, the determinant of $i \times i$ minors is preserved modulo $\mathfrak{g}$. Let $M_i \subseteq \mathcal{O}_K^{n-i}$ be the $\mathcal{O}_K$-module defined by

$$\mathfrak{a}_1(a_{1,n-i}, \cdots, a_{1,n}) + \cdots + \mathfrak{a}_n(a_{n,n-i}, \cdots, a_{n,n}),$$

and $\mathfrak{g}(M_i)$ its determinantal ideal. The operations performed at Step 6 to 10 in Algorithm 3 preserve $\mathfrak{g}(M_i)$ while after Step 11, our pseudo-basis $[(B_i)_{i\leq n}, (\mathfrak{b}_i)_{i\leq n}]$ only defines a module $M' \subseteq \mathcal{O}_K^n$ satisfying

$$\mathfrak{g}(M_i') + \mathfrak{g} = \mathfrak{g}(M_i) + \mathfrak{g}.$$

This property is the equivalent of the integer case when the HNF is taken modulo a multiple D of the determinant of the lattice. To recover the ideals $\mathfrak{c}_i$ of a pseudo-HNF of M, we first notice that

$$\begin{aligned}\forall i, \mathfrak{g}(M_i') + \mathfrak{g} = \mathfrak{g}(M_i) + g &= \mathfrak{c}_{n-i}\cdots\mathfrak{c}_n + \mathfrak{g}\\ &= \mathfrak{c}_{n-i}\cdots\mathfrak{c}_n + \mathfrak{c}_1\cdots\mathfrak{c}_n\\ &= \mathfrak{c}_{n-i}\cdots\mathfrak{c}_n.\end{aligned}$$

On the other hand, $\mathfrak{g}(M_i') + \mathfrak{g} = \mathfrak{b}_{n-i}\cdots\mathfrak{b}_n + \mathfrak{g}$. Thus, we have

$$\forall i,\ \mathfrak{b}_{n-i}\cdots\mathfrak{b}_n + \mathfrak{g} = \mathfrak{c}_{n-i}\cdots\mathfrak{c}_n,$$

which allows us to recursively recover the $\mathfrak{c}_i$ from the $(\mathfrak{b}_j)_{j\geq i}$ and $\mathfrak{g}$. Indeed, as in the integer case, it boils down to taking

$$\mathfrak{c}_i = \frac{\mathfrak{g}}{\prod_{j>i}\mathfrak{c}_j} + \mathfrak{b}_i.$$

To do so, we keep track of $\mathfrak{g}_i := \frac{\mathfrak{g}}{\prod_{j>i}\mathfrak{c}_j}$ throughout Algorithm 4 that reconstructs the actual pseudo-HNF from its modular version given by Algorithm 3. At each step we set

$$\mathfrak{c}_i \leftarrow \mathfrak{b}_i + \mathfrak{g}_i.$$

This replacement of the ideals in the pseudo-basis defining our module impacts the corresponding vectors in K^n as well. In particular, we require that the diagonal elements all be 1. Do ensure thus, we find $u \in \mathfrak{b}_i\mathfrak{c}_i^{-1}$, $v \in \mathfrak{g}_i\mathfrak{c}_i^{-1}$ such that $u + v = 1$ which implies that

$$\mathfrak{c}_i(uB_i + ve_i) \subseteq \mathfrak{b}_iB_i + \mathfrak{g}_ie_i,$$

where the i-th coefficient of $uB_i + ve_i \in K^n$ is 1 and the coefficient of index $j > i$ in $uB_i + ve_i$ are 0. Then we set

$$W_i \leftarrow uB_i \bmod \mathfrak{g}_i\mathfrak{c}_i^{-1},$$

and observe that $\sum_i \mathfrak{c}_iW_i \subseteq M$. These $\mathcal{O}_K$-modules have the same determinantal ideal, and as in the integer case, we can prove that it is sufficient to ensure that they are equal.

$uB_i + ve_i = W_i + d_i$ where the coefficients of $d_i \in (\mathfrak{g}_i/\mathfrak{c}_i)^n$ of index $j > i$ are 0. The vector d_i satisfies $\mathfrak{c}_id_i \subseteq \mathfrak{g}_id_i'$ where $d_i' \in \mathcal{O}_K^n$ with coefficients $j > i$ equal to 0. This allows us to state that

$$\mathfrak{c}_iW_i \subseteq \mathfrak{b}_iB_i + \mathfrak{g}_ie_i + \mathfrak{c}_id_i \subseteq M + \mathfrak{g}_ie_i + \mathfrak{g}_id_i' \subseteq M + \mathfrak{g}_iD_i,$$

where the coefficients of $D_i \in \mathcal{O}_K^n$ of index $j > i$ equal 0. We now want to prove that $\mathfrak{c}_iW_i \subseteq M$. To do this, we prove that $\mathfrak{g}_iD_i \subseteq M$.

LEMMA 1. *Let $M = \mathfrak{a}_1 A_1 + \cdots \mathfrak{a}_n A_n \in \mathcal{O}_K^n$, then we have*

$$\mathfrak{g}(M)\mathcal{O}_K^n \subseteq M$$

PROOF. We can prove by induction that if $[(B_i), (\mathfrak{b}_i)]$ is a pseudo-HNF basis of M, then

$$\forall i,\ \mathfrak{g}_1 \cdots \mathfrak{g}_i e_i \subseteq M,$$

where e_i is the i-th vector of the canonical basis of $\mathcal{O}_K^n$. Our statement immediatly follows. □

We now consider the intersection N_i of our module $M \subseteq \mathcal{O}_K^n$ with $\mathcal{O}_K^i$. Note that with the previous definitions, we have in particular $M = N_i \oplus M_i$.

LEMMA 2. *Let $i \leq n$ and $D \in \mathcal{O}_K^n$ a vector whose entries of index $j > i$ are 0. Then we have*

$$\mathfrak{g}_i D \subseteq M.$$

PROOF. From Lemma 1, we know that $\mathfrak{g}_i \mathcal{O}_K^i \subseteq N_i$. If $D_i \in \mathcal{O}_K^i$ is the first i coordinates of D, then $\mathfrak{g}_i D_i \subseteq N_i$, and as the last $n - i$ coordinates of D are 0, we have

$$\mathfrak{g}_i D \subseteq M.$$

□

The module generated by the pseudo-basis $[(W_i), (\mathfrak{c}_i)]$ computed by Algorithm 4 is a subset of M. We ensured that its determinantal ideal $\prod_i \mathfrak{c}_i$ equals the determinantal ideal $\mathfrak{g}$ of M. Let us prove that it is sufficient to ensure that

$$\mathfrak{c}_1 W_1 + \cdots + \mathfrak{c}_n W_n = M.$$

LEMMA 3. *Let $M = \sum_{i \leq n} \mathfrak{a}_i A_i$ and $M' = \sum_{i \leq n} \mathfrak{b}_i B_i$ two n-dimensional $\mathcal{O}_K$-modules such that $M' \subseteq M$ and $\mathfrak{g}(M') = \mathfrak{g}(M)$. Then necessarily*

$$M = M'.$$

PROOF. Let $[(W_i), (\mathfrak{c}_i)]$ be a pseudo-HNF for M, and $[(W_i'), (\mathfrak{c}_i')]$ a pseudo-HNF for M'. By assumption, we have $\prod_i \mathfrak{c}_i = \prod_i \mathfrak{c}_i'$, and $M' \subseteq M$. As both matrices W and W' have a lower triangular shape, it is clear that

$$\forall i,\ \sum_{j \leq i} \mathfrak{c}_j' W_j' \subseteq \sum_{j \leq i} \mathfrak{c}_j W_j. \tag{4}$$

As the diagonal coefficients of both W and W' are 1, we see by looking at the inclusion in the coefficient i of (4) that $\mathfrak{c}_i' \subseteq \mathfrak{c}_i$. Then as $\mathfrak{g}(M) = \mathfrak{g}(M')$, we have

$$\forall i \mathfrak{c}_i = \mathfrak{c}_i'.$$

Now let us prove by induction that

$$\forall i,\ \mathfrak{c}_i W_i \subseteq \mathfrak{c}_1 W_1' + \cdots + \mathfrak{c}_i W_i'. \tag{5}$$

This assertion is clear for $i = 1$ since $W_1 = W_1' = e_1$. Then, assuming (5) for $1, \cdots, i-1$, we first use the fact that

$$\mathfrak{c}_i W_i' \subseteq \mathfrak{c}_1 W_1 + \cdots + \mathfrak{c}_i W_i.$$

In other words, $\forall c_i' \in \mathfrak{c}_i$, $\exists (c_1, \cdots, c_i) \in \mathfrak{c}_1 \times \cdots \times \mathfrak{c}_i$ such that

$$c_i'(w_{i,1}', \cdots, w_{i,i-1}', 1) = \left(\sum_{1 \leq j \leq i} c_j w_{j,1}, \cdots, c_i w_{i,i-1} + c_{i-1}, c_i \right).$$

In particular, $c_i = c_i'$, which allows us to state that $\forall c_i \in \mathfrak{c}_i$, $\exists (c_1, \cdots, c_{i-1}) \in \mathfrak{c}_1 \times \cdots \times \mathfrak{c}_{i-1}$ such that

$$\begin{aligned} c_i w_{i,i-1} &= c_{i-1} + c_i w_{i,i-1}' \\ c_i w_{i,i-2} &= c_{i-2} + c_{i-1} w_{i-1,i-2} + c_i w_{i,i-2}' \\ \vdots &= \vdots \\ c_i w_{i,1} &= c_1 + \cdots + c_{i-1} w_{i-1,1} + c_i w_{i,1}'. \end{aligned}$$

This shows that

$$\mathfrak{c}_i W_i \subseteq \mathfrak{c}_1 W_1 + \cdots + \mathfrak{c}_{i-1} W_{i-1} + \mathfrak{c}_i W_i',$$

and since we have $\forall j < i,\ \mathfrak{c}_j W_i \subseteq \sum_{j<i} \mathfrak{c}_j W_j'$, we obtain the desired result. □

Lemma 3 is a generalization of the standard result on $\mathbb{Z}$-modules stating that if $L' \subseteq L$ and $\det(L) = \det(L')$, then $L = L'$. Although implied in [2, Chap. 1], Lemma 3 is not stated, nor proved in the litterature. Yet, it is essential to ensure the validity of Algorithm 4.

PROPOSITION 9. *The $\mathcal{O}_K$-module defined by the pseudo-basis $[(W_i), (\mathfrak{c}_i)]$ obtained by applying lgorithm 4 to the pseudo-HNF of M modulo $\mathfrak{g}(M)$ satisfies*

$$\mathfrak{c}_1 W_1 + \cdots + \mathfrak{c}_n W_n = M.$$